MAT976 - Group LASSO

Trent Guy Lemkus

7 May 2018

Contents

1 Document Updated last:

```
stTime <- Sys.time()
stTime
```

```
## [1] "2018-04-29 23:00:40 EDT"
```

2 Abstract

In 2006, Yuan and Lin introduced the group lasso in order to allow predefined groups of covariates to be selected into or out of a model together, so that all the members of a particular group are either included or not included. While there are many settings in which this is useful, perhaps the most obvious is when levels of a categorical variable are coded as a collection of binary covariates. In this case, it often doesn't make sense to include only a few levels of the covariate; the group lasso can ensure that all the variables encoding the categorical covariate are either included or excluded from the model together.

The idea behind this report is to not only delve into the mathematical details of the Grouped LASSO, but also simplify the details so that the average reader can walk away with an approximate understanding of this particular derivation of the LASSO. The traditional LASSO is amenable to data sets where sparsity is more likely to uncover the "true" model, but a downfall of this method is its inability to perform well when the design matrix contains much multicollinearity. In such an event we often find the L1 regularization to not only produce a sub-standard model, but it will also fail at obtaining maximal sparsity or any as is seen in the case of the Sonar data set we introduce in a later section. Furthermore, we make use of the Sonar data set to illustrate how the application of the Grouped LASSO can ameliorate the formerly mentioned sparsity issue in the light of multicollinearity.

The mathematics and formulae behind the Grouped LASSO are discussed, but due to it's complex nature and even more complex solution path we have added a small example to illustrate the inner workings of this derivation. We also discuss how to group your predictors based on correlation as well as the interesting result that sparsity is not only obtained, but the algorithm has an ability to do so by predictor within a group and not by eliminating entire groups. We do, however, avoid discussions about the variants of the Group LASSO: Sparse Group LASSO, Overlap Group LASSO, Sparse GAM with Group LASSO etc.

3 The Grouped LASSO

There are many regression problems in which the covariates have a natural group structure, and it is desirable to have all coefficients within a group become nonzero (or zero) simultaneously. The various forms of group lasso penalty are designed for such situations. We first define the group lasso and then develop this and other motivating examples.
The setup is as follows:

There are J groups where $j = 1, ..., J$ The vector $Z_j \in \mathbb{R}^{P_j}$ represents the covariates in group j

The Goal:

To predict a real-values response $Y \in \mathbb{R}$ based on the collection of covariates in our J groups $(Z_1, ..., Z_j)$

3.1 Defining the Linear Model

The linear model can be defined as:

$$\mathbb{E}(Y|Z) = \theta_0 + \sum_{j=1}^{J} Z_j^T \theta_j, \ \textit{where } \theta_j \in \mathbb{R}^{P_j}$$

Note: θ_j represents a group of p_j regression coefficients.

3.2 Defining The Convex Problem

Given a collection of N samples $\{(y_i, z_{i1}, z_{i2}, ..., z_{i,J})\}_{i=1}^{N}$ the Group LASSO solves the following covex problem:

$$\underset{\theta_0 \in \mathbb{R}, \theta_j \in \mathbb{R}^{p_j}}{\text{minimize}} \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \theta_0 - \sum_{j=1}^{J} z_{ij}^T \theta_j)^2 + \lambda \sum_{j=1}^{J} \|\theta_j\|_2 \right\}$$

Where $\|\theta_j\|_2$ is the Euclidean norm of the vector θ_j and the following hold true:

- depending on $\lambda \geq 0$, either the entire vector $\hat{\theta_j}$ will be zero, or all its elements will be nonzero.
- when $p_j = 1$, then we have $\|\theta_j\|_2 = |\theta_j|$, so if all the groups are singletons i.e. every group represents a single predictor, than the optimization problem reduces to the ordinary LASSO.
- All groups are equally penalized, a choice which leads larger groups to be more likely to be selected. In their original proposal, Yuan and Lin (2006)[1] recommended weighting the penalties for each group according to their size, by a factor $\sqrt{p_j}$. One could also argue for a factor of $\|\mathbb{Z}_j\|_F$ where the matrices are not orthonormal.

Here we compare the constraint region for the Group LASSO (left) to that of the LASSO in $\mathbb{R}^3$. We see that the Group LASSO shares attributes of both the l_2 and l_1 balls:

My Folder

4 Multi-Level Sparsity: Sparse Group LASSO

I would be remiss if I didn't, at the very least, discuss the Sparse Group LASSO, which takes the Group LASSO a step further by not only imposing sparsity on the groups, but also selects which coefficients are non-zero within the groups. From a technical standpoint this is vital considering the core uncertainty in this style of problem if that of determining the groups. If you have selected your groups to include, even just one, important variable(s) then this coefficient would be shrunk to zero along with all other coefficients in said group, however with the advantages of using the Sparse Group LASSO there is a strong chance that "important" coefficients within zeroed groups may be recovered in the final model.

In order to achieve within group sparsity, we augment the basic Group LASSO with an additional l_1-penalty, leading to the convex program:

$$\underset{\{\theta_j \in \mathbb{R}^{p_j}\}}{\text{minimize}} \left\{ \frac{1}{2} \|\mathbf{y} - \sum_{j=1}^{J} (Z)_j \theta_j\|_2^2 + \lambda \sum_{j=1}^{J} [(1-\alpha)\|\theta_j\|_2 + \alpha \|\theta_j\|_1] \right\}$$

with $\alpha \in [0, 1]$. Much like the Elastic Net, the parameter α creates a bridge between the Group LASSO ($\alpha = 0$) and the LASSO ($\alpha = 1$). Below is the image that contrasts the Group LASSO constraint region with that of the Sparse Group LASSO for the case in $\mathbb{R}^3$:

My Folder

Note: in the two horizontal planes the constraint region resembles that of the elastic net being more rounded than angular.

5 The Sonar Data Set

5.1 Description

This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network [2]. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.

Each pattern is a set of 60 numbers in the range 0.0 to 1.0 [3]. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp.

The label associated with each record contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder). The numbers in the labels are in increasing order of aspect angle, but they do not encode the angle directly.

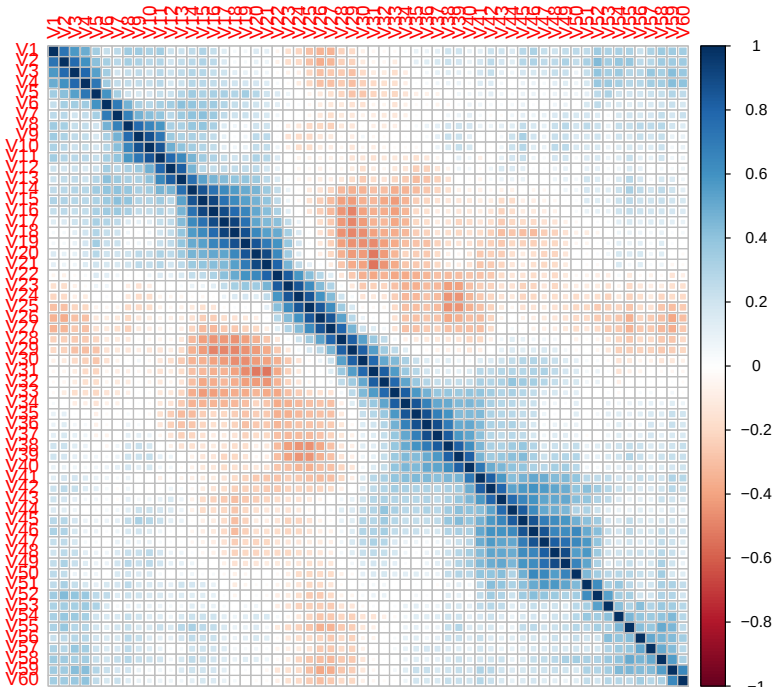
Below is a preview of the data set:

```
## 'data.frame':    208 obs. of  61 variables:
##  $ V1   : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223 0.0164 ...
##  $ V2   : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0375 0.0173 ...
##  $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
##  $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
##  $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
##  $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
##  $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
##  $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
##  $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
##  $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
##  $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
##  $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
##  $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
##  $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
##  $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
##  $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
##  $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
##  $ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
##  $ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
##  $ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
##  $ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
##  $ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
##  $ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
##  $ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
##  $ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
##  $ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
##  $ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
##  $ V28  : num  0.808 0.277 0.853 0.252 1 ...
##  $ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
##  $ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
##  $ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
##  $ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
##  $ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
##  $ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
##  $ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
##  $ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
```

```
##  $ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
##  $ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
##  $ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
##  $ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
##  $ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
##  $ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
##  $ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
##  $ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
##  $ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
##  $ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
##  $ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
##  $ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
##  $ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.0777 0.0092 ...
##  $ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0439 0.0198 ...
##  $ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0061 0.0118 ...
##  $ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0145 0.009 ...
##  $ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.0128 0.0223 ...
##  $ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.0145 0.0179 ...
##  $ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058 0.0084 ...
##  $ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0049 0.0068 ...
##  $ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065 0.0032 ...
##  $ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0093 0.0035 ...
##  $ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.0059 0.0056 ...
##  $ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0022 0.004 ...
##  $ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 2 ...
```

5.2 Why Use A Group LASSO?

The primary issue with the Sonar data is the eggregious multicollinearity present. We display this with a correlation heat map below:

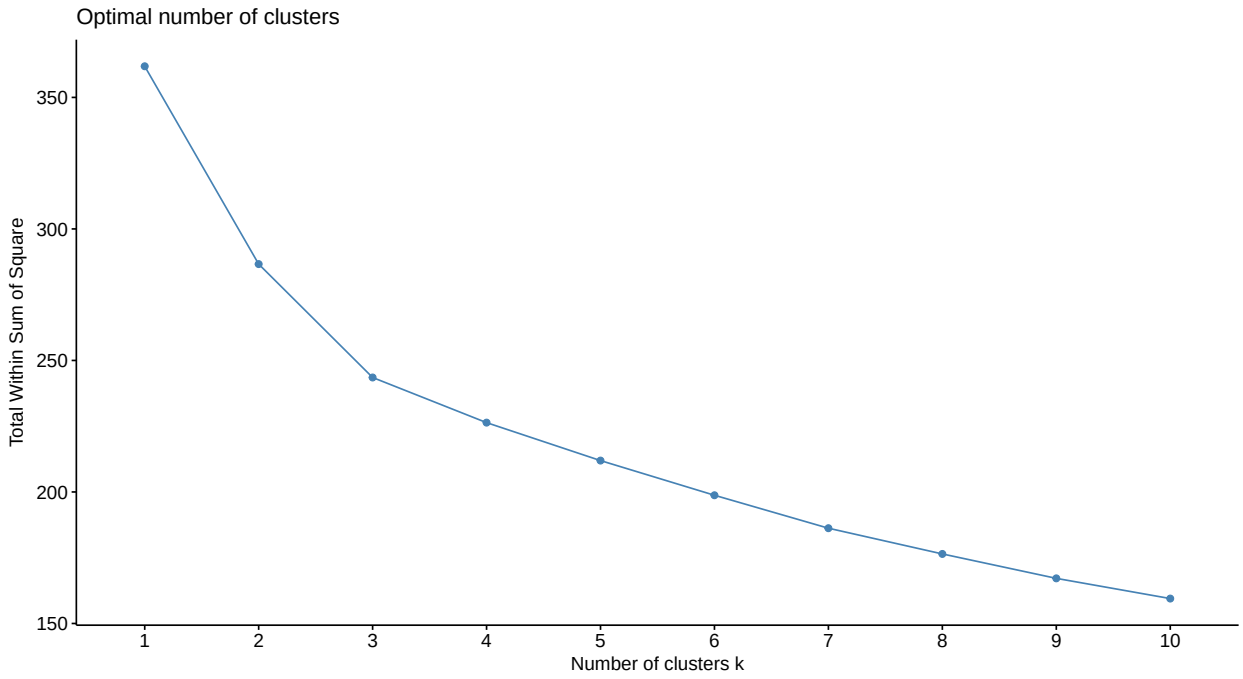


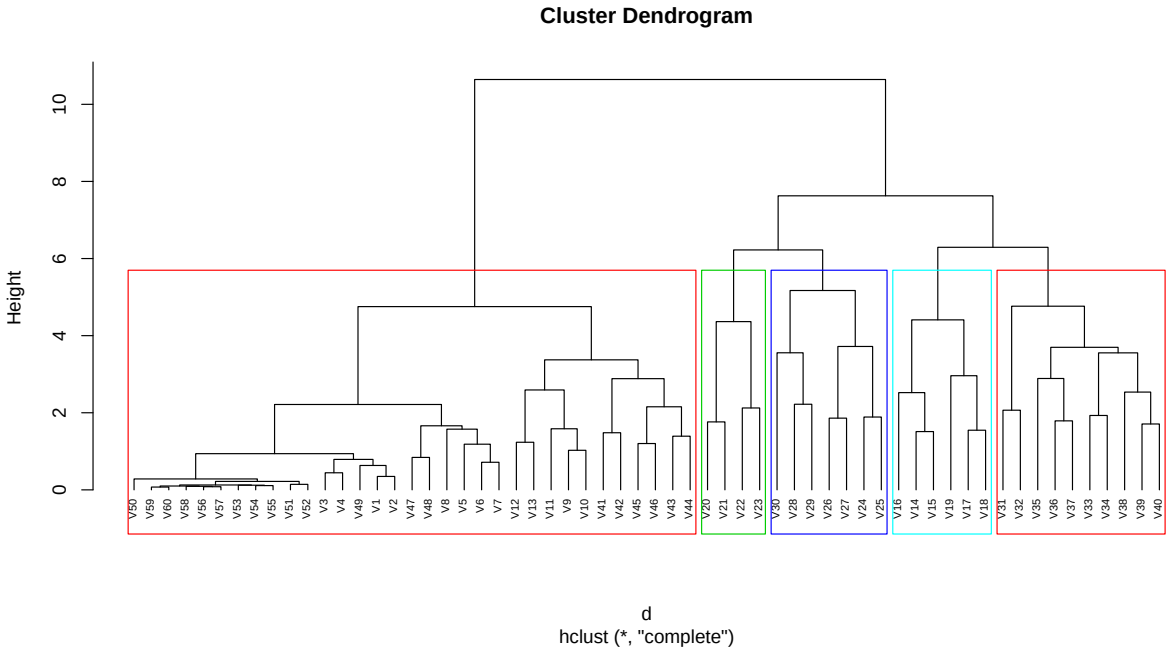
As we can see from the above correlation heat map, we have thick clusters of predictors that are highly correlated with one another. This is an issue when trying to predict the response with the LASSO. Our results would typically contain high variability and almost no sparsity is obtained, but this is almost single handedly ameliorated with the Group LASSO.

5.3 How to Determine Groups?

There are various clustering algorithms that can be utilized when forming groups: The method that seemed to work the best is Hierarchical Clustering with a Euclidean distance and a Complete linkage that I determined through trial and error, however I did investigate a contemporary clustering algorithm called DBSCAN (Density Based Clustering of Applications with Noise). Because this report is focused on the Group LASSO I will ommit the more techinical details, but I will state that it did not perform well because the groups are too sparse in the PC1/PC2 dimension.

Below we apply the "elbow rule" to the Scree Plot and highlight the 5 selected groups on the Dendrogram:





5.3.1 Train Test Split

We split the data set into a 90/10 train-test split. The main reason for this type of split is purely because the number of samples is disproportionate to the number of observations. The training set is left containing 3 samples for every predictor. This is far below the recomended rule of thumb of 10 (for Regression problems), albeit an area where the LASSO algorithm shines.

5.4 Implimented Models

We fit and compare the following models:

1) Multi-Layer Perceptron (MLP)

The first model I chose simply because the developers of the Sonar data set (Gorman & Sejnowski 1988)[2] fit a MLP model to this data with perfect prediction accuracy. I have very little doubt that the training took an immense amount of time with many iterations to the number of layers and included neurons. I made use of the SNNS (Stuttgart Neural Network Simulator) package in R. We tuned this model using a tuning grid for three layers with three settings of 10, 20, and 30 hidden neurons. Due to time constraints a larger tuning grid was unable to be explored.

2) LASSO

This model was fit using 10 fold 5 repeated cross validation strategy within the framework of the "Caret" package. I did make use of a tuning grid for λ.

3) Elastic Net

This model was fit using 10 fold 5 repeated cross validation strategy within the framework of the "Caret" package. I did make use of a tuning grid for α and λ.

4) Group LASSO

This model was fit using a cross validation function within the "gglasso" package with 5 folds as an option to balance the CV settings of the LASSO/Elastic Net combination with the Group LASSO/Group Ridge combination for comparability.

5) Group Ridge

This model was fit using a cross validation function within the "gglasso" package with 5 folds as an option to balance the CV settings of the LASSO/Elastic Net combination with the Group LASSO/Group Ridge combination for comparability.

The comparison will be made based on test sample prediction accuracy, time elapsed, and model sparsity.

5.5 Model Comparison

5.5.1 Time Comparison

Below is a table comparing all 5 models:

Model	Accuracy	Run Time	No. of Predictors
MLP	0.45	7.15	60
LASSO	0.75	21.18	54
Elastic Net	0.7	3.39	60
Group LASSO	0.65	41.17	50
Group Ridge	0.7	39.87	60

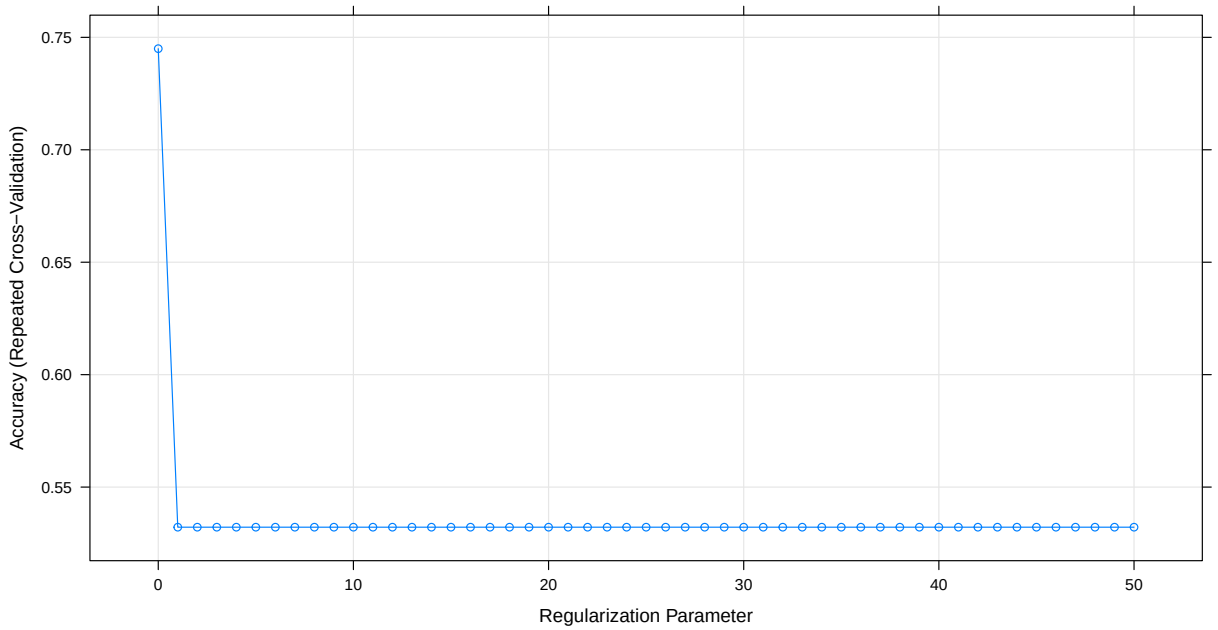
Analysis

The outright winner if run time is the exclusive concern would typically be the Elastic Net and the worst run time is usually the Group LASSO, but was narrowly beat out by the Group Ridge.

5.5.2 Sparsity Comparison

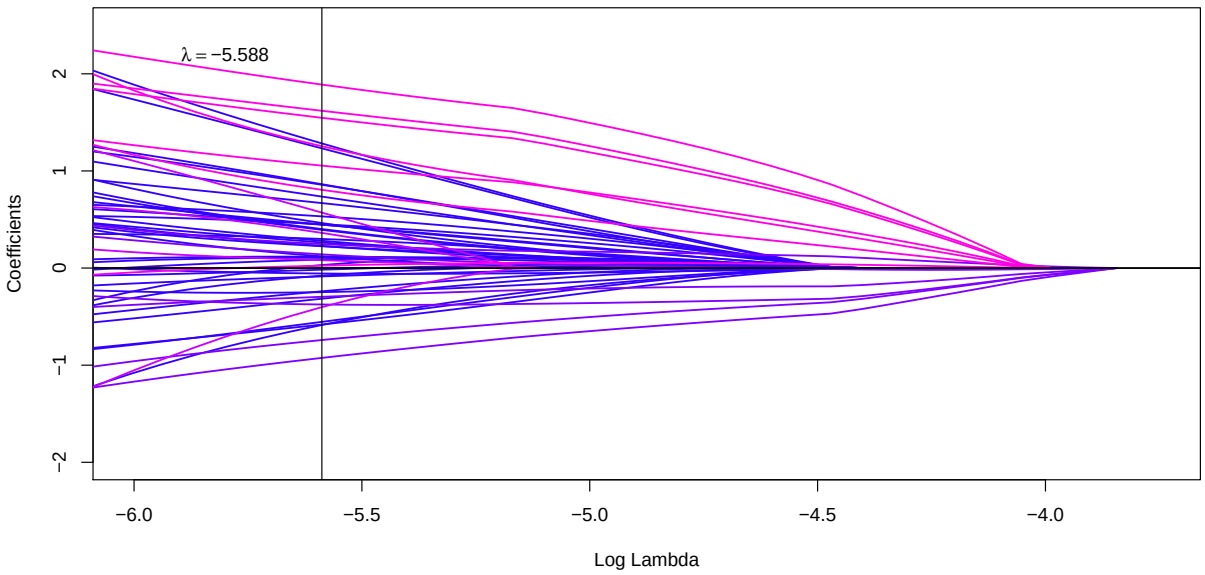
We omit the MLP and the Group Ridge from this discussion since they are not sparse models to begin with

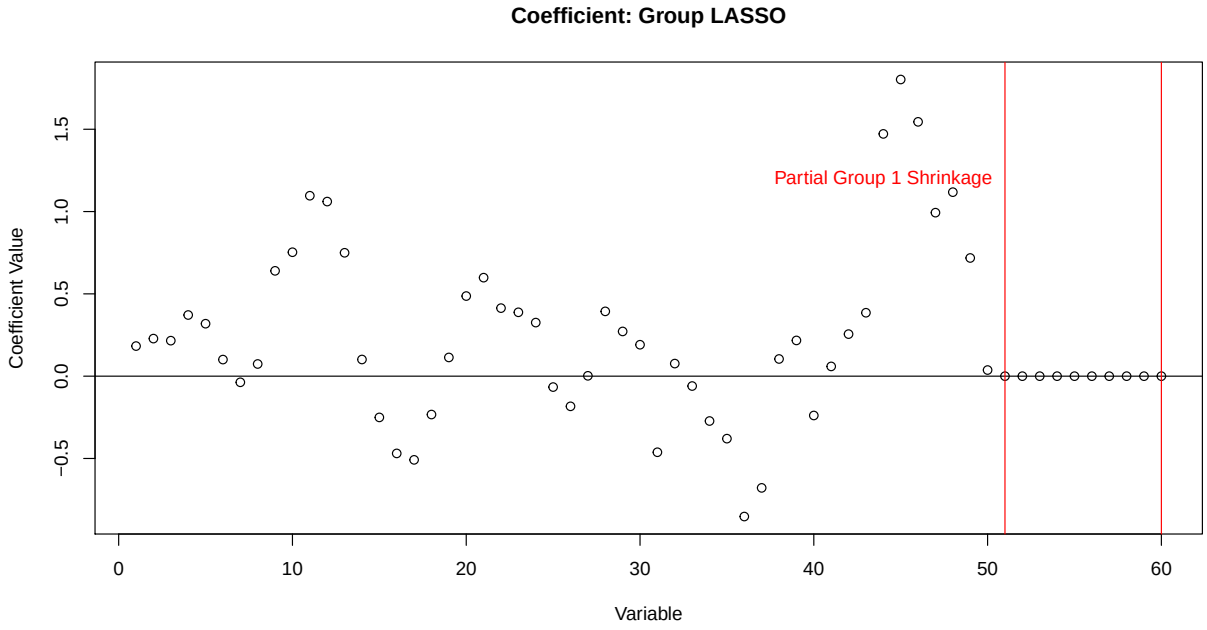
LASSO



The accuracy of the LASSO model is highest when there is no coefficient shrinkage. This is to be expected from the multicollinearity issue we had previously assessed and described above.

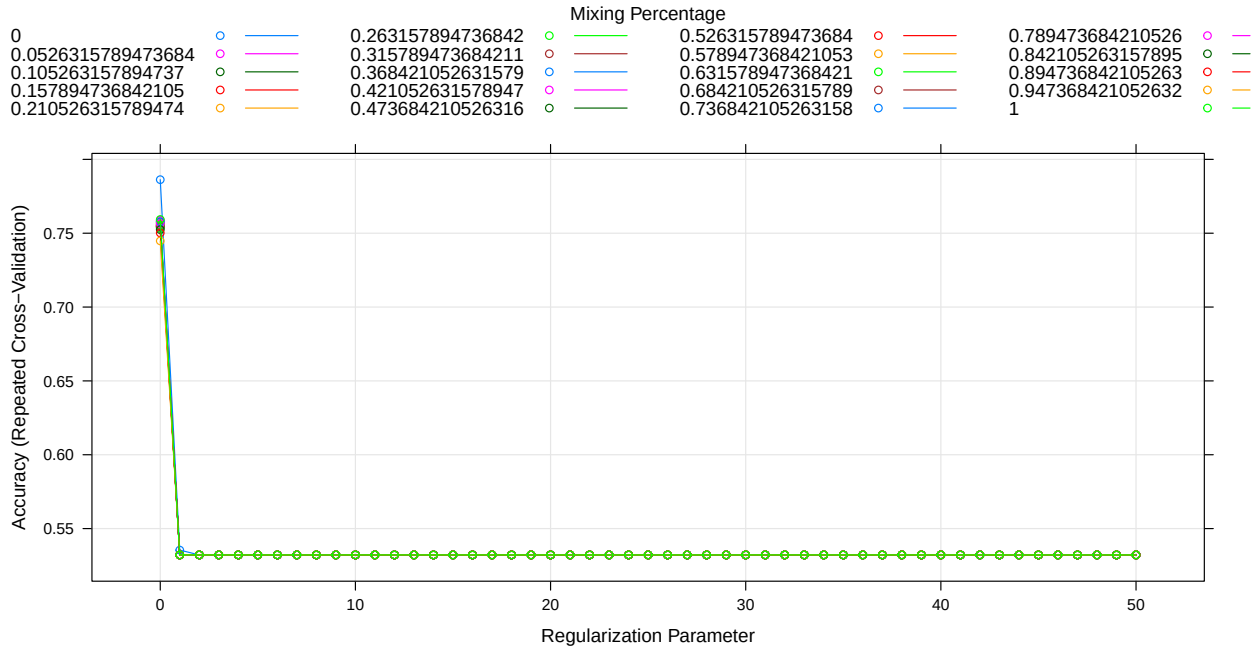
Group LASSO





Here we see that not only was there shrinkage but there were 10 coefficients shrunk to zero, however these were all coefficients that belonged to Group 1, but did not encompass the full membership of Group 1 leading me to believe that the package Glasso impliments the Sparse Group LASSO and not the Group LASSO singularly.

Elastic NET



Once again we see a similar plot to that of the LASSO where all the repeated CV attempts achieved highest accuracy with no parameter shrinkage.

5.5.3 Prediction Accuracy

MLP

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M  0  0
##          R 11  9
##
##                Accuracy : 0.45
##                  95% CI : (0.2306, 0.6847)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 0.869235
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 0.002569
##
##             Sensitivity : 0.00
##             Specificity : 1.00
##          Pos Pred Value :  NaN
##          Neg Pred Value : 0.45
##              Prevalence : 0.55
##          Detection Rate : 0.00
##    Detection Prevalence : 0.00
##       Balanced Accuracy : 0.50
##
##        'Positive' Class : M
##
```

LASSO

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction M R
##          M 7 1
##          R 4 8
##
##                Accuracy : 0.75
##                  95% CI : (0.509, 0.9134)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 0.05533
##
##                   Kappa : 0.5098
##  Mcnemar's Test P-Value : 0.37109
##
##             Sensitivity : 0.6364
##             Specificity : 0.8889
##          Pos Pred Value : 0.8750
##          Neg Pred Value : 0.6667
##              Prevalence : 0.5500
##          Detection Rate : 0.3500
##    Detection Prevalence : 0.4000
```

```
##       Balanced Accuracy : 0.7626
##
##        'Positive' Class : M
##
```

Eslastic Net

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction M R
##          M 7 2
##          R 4 7
##
##                Accuracy : 0.7
##                  95% CI : (0.4572, 0.8811)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 0.1299
##
##                   Kappa : 0.4059
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.6364
##             Specificity : 0.7778
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.6364
##              Prevalence : 0.5500
##          Detection Rate : 0.3500
##    Detection Prevalence : 0.4500
##       Balanced Accuracy : 0.7071
##
##        'Positive' Class : M
##
```

Group LASSO

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction M R
##          M 6 2
##          R 5 7
##
##                Accuracy : 0.65
##                  95% CI : (0.4078, 0.8461)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 0.2520
##
##                   Kappa : 0.3137
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.5455
##             Specificity : 0.7778
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.5833
##              Prevalence : 0.5500
```

```
##          Detection Rate : 0.3000
##    Detection Prevalence : 0.4000
##       Balanced Accuracy : 0.6616
##
##        'Positive' Class : M
##
```

Ridge Group

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction M R
##          M 6 1
##          R 5 8
##
##                Accuracy : 0.7
##                  95% CI : (0.4572, 0.8811)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 0.1299
##
##                   Kappa : 0.4175
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.5455
##             Specificity : 0.8889
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.6154
##              Prevalence : 0.5500
##          Detection Rate : 0.3000
##    Detection Prevalence : 0.3500
##       Balanced Accuracy : 0.7172
##
##        'Positive' Class : M
##
```

6 Conclusion

The Group LASSO not only achieves sparsity, but it does so with the highest accuracy, however one should be cautious of the time taken to run since this algorithm becomes increasingly slow with the increase in predictors and is further slowed down when running combining methods like Cross Validation.

7 References

[1] Yuan, M. and Lin, Y. (2006) "Model Selection and Estimation in Regression with Grouped Variables". *J. R. Statist. Soc.* B, 68, 49-67

[2] Hastie. H, Tibshirani. R, and Wainwright, M. (2016) "Statistical Learning With Sparsity: The Lasso and Generalizations". *CRC Press*

[3] Gorman, R. P., and Sejnowski, T. J. (1988). "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" in Neural Networks, Vol. 1, pp. 75-89.

[4] Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learn-

ing databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

7.1 R Code for Report

```
stTime <- Sys.time()
stTime
# Load Packages
pkg <- c("ggplot2", "knitr", "markdown",
         "glmnet", "caret", "gglasso",
         "corrplot","RColorBrewer","zoo",
         "factoextra", "mlbench", "rlang")

sapply(pkg, require, character.only = TRUE)

# Knitr Options
knitr::opts_chunk$set(progress = TRUE,
                      fig.width=11,
                      fig.height=6,
                      echo = FALSE,
                      message = FALSE,
                      warning = FALSE,
                      fig.align = "center",
                      cache = TRUE,
                      cache.lazy = TRUE,
                      tidy=TRUE,
                      tidy.opts=list(blank=TRUE, width.cutoff=65) )


# Setup logical directory structure
ROOT.DIR <- "C:\\Users\\Admin\\Desktop\\UNH\\Classes\\TOPICS - SPRING 2018\\Final Report - The Grouped
DATA.DIR <- paste(ROOT.DIR,"data",sep="/")
# Import Data
data(Sonar)

# Investigate Sonar Data Set
str(Sonar)
# reomve rows with missing values
Sonar = Sonar[complete.cases(Sonar),]
Sonar = Sonar[sample(nrow(Sonar)),]

# Assign Predictors and Response
y = Sonar$Class
X = Sonar[,-61]

# Correlation Plot
# create correlation matrix
cor_mat=cor(X, use="complete.obs")
# plot cor matrix
corrplot(cor_mat,
         order = "original",
         method="square")
# Apply Hierarchical Clustering
```

```r
# Dissimilarity matrix
d = dist(t(X), method = "euclidean")

# Hierarchical clustering using Complete Linkage
hc1 = hclust(d, method = "complete" )

# Apply Elbow rule
fviz_nbclust(X, FUN = hcut, method = "wss") # It seems 4 - 6 groups may be appropriate

# Plot dendrogram with 5 selected groups
plot(hc1, cex = 0.6, hang = -1)
rect.hclust(hc1, k = 5, border = 2:5)

# We can see that 5 groups is a reasonable choice

# Create groups for Group LASSO
# Cut tree into 5 groups
sub_grp = cutree(hc1, k = 5)
grp=c(as.matrix(sub_grp))
# Define Groups
grp1 = X[,names(which(sub_grp == 1))]
grp2 = X[,names(which(sub_grp == 2))]
grp3 = X[,names(which(sub_grp == 3))]
grp4 = X[,names(which(sub_grp == 4))]
grp5 = X[,names(which(sub_grp == 5))]
# Create a 90/10 Train Test Split
set.seed(777)
trainIndex <- createDataPartition(y, p = .9,
                                  list = FALSE,
                                  times = 1)
# Seperate Train and Test Sets
y_train = y[trainIndex]
y_test = y[-trainIndex]
X_train = X[trainIndex,]
X_test = X[-trainIndex,]
########################################
## Multi-Layer Perceptron
########################################
# Define Grid
layer1 = seq(10,30,10)
layer2 = seq(10,30,10)
layer3 = seq(10,30,10)
mlp_grid = expand.grid(.layer1 = layer1,
                       .layer2 = layer2,
                       .layer3 = layer3)

# Train Model
set.seed(1)
start.time.mlp <- Sys.time()
fit.mlp = caret::train(x = data.matrix(X_train),
                y = y_train,
                method = "mlpML",
                #trControl = trnCtrl,
```

```
                #act.fct = 'logistic',
                tuneGrid = mlp_grid,
                standardize = FALSE)
end.time.mlp <- Sys.time()
time.taken.mlp <- end.time.mlp - start.time.mlp
# Plot and Model Details
#plot(my.train)
best.params = fit.mlp$bestTune
my.mlp.model <- fit.mlp$finalModel
# Prediction on Test Set
pred.mlp = predict(fit.mlp, newdata = data.matrix(X_test))
mlp_matrix = confusionMatrix(pred.mlp, y_test)
########################################
## LASSO
########################################
# Define Grid
lambda.grid <- seq(0, 50)
alpha.grid <- seq(0, 1, length = 20)

srchGrd = expand.grid(.alpha = 1, .lambda = lambda.grid)

# Setup CV Function
trnCtrl = trainControl(
  method = "repeatedCV",
  number = 10,
  repeats = 5)

# Train Model
set.seed(2)
start.time.L <- Sys.time()
fit.LASSO <- caret::train(x = data.matrix(X_train),
                  y = y_train,
                  method = "glmnet",
                  tuneGrid = srchGrd,
                  trControl = trnCtrl,
                  standardize = FALSE)
end.time.L <- Sys.time()
time.taken.L <- end.time.L - start.time.L
# Plot and Model Details
best.params = fit.LASSO$bestTune
my.LASSO.model <- fit.LASSO$finalModel
# Prediction on Test Set
pred.LASSO = predict(fit.LASSO, newdata = data.matrix(X_test))
LASSO_matrix = confusionMatrix(pred.LASSO, y_test)
########################################
## Elastic Net (tuned alpha)
########################################
# Define Grid
lambda.grid <- seq(0, 50)
alpha.grid <- seq(0, 1, length = 20)

srchGrd = expand.grid(.alpha = alpha.grid, .lambda = lambda.grid)

# Setup CV Function
```

```
trnCtrl = trainControl(
  method = "repeatedCV",
  number = 10,
  repeats = 5)

# Train Model
set.seed(3)
start.time.EN <- Sys.time()
fit.glmN <- train(x = data.matrix(X_train),
                  y = y_train,
                  method = "glmnet",
                  tuneGrid = srchGrd,
                  trControl = trnCtrl,
                  standardize = FALSE)
end.time.EN <- Sys.time()
time.taken.EN <- end.time.EN - start.time.EN
# Plot and Model Details
best.params = fit.glmN$bestTune
my.glmnet.model <- fit.glmN$finalModel
# Prediction on Test Set
pred.glmnet = predict(fit.glmN, newdata = data.matrix(X_test))
glmN_matrix = confusionMatrix(pred.glmnet, y_test)
########################################
## Group LASSO
########################################
# Cross Validation (This can take a substantial amount of time - around 45 mins)
X_train = as.matrix(X_train)
y_train = ifelse(y_train == "M", 1 , -1) # Binary Factor needs to be numeric {-1,1} format
set.seed(4)
start.time.GLASSO <- Sys.time()
fit.GLASSO=cv.gglasso(x=X_train,y=y_train,group=grp,
                      loss="logit",
                      pred.loss="L1", # Penalized Logistic Regression
                      nfolds=10)
end.time.GLASSO <- Sys.time()
time.taken.GLASSO <- end.time.GLASSO - start.time.GLASSO

# Prediction
pred.gglasso = predict(fit.GLASSO, newx = data.matrix(X_test),
                       s = "lambda.min", type = "class")
pred.gglasso = ifelse(pred.gglasso == 1, "M", "R")
pred.gglasso = as.factor(pred.gglasso)
GLASSO_matrix = confusionMatrix(pred.gglasso, y_test)
########################################
# Group Ridge
########################################
# Cross Validation (This can take 40 - 45 minutes)
set.seed(5)
start.time.RLASSO <- Sys.time()
fit.cv.ridge=cv.gglasso(x=X_train,y=y_train,group=grp,
                        loss="logit",
                        pred.loss="L2", # Penalized Logistic Regression
                        nfolds=10)
end.time.RLASSO <- Sys.time()
```

```r
time.taken.RLASSO <- end.time.RLASSO - start.time.RLASSO

# Best Lambda
lmbda=fit.cv.ridge$lambda.min

# Prediction
pred.gglasso.L2 = predict(fit.cv.ridge, newx = data.matrix(X_test),
                          s = "lambda.min", type = "class")
pred.gglasso.L2 = ifelse(pred.gglasso.L2 == 1, "M", "R")
pred.gglasso.L2 = as.factor(pred.gglasso.L2)
RGroup_matrix = confusionMatrix(pred.gglasso.L2, y_test)
# LASSO CV Plot
plot(fit.LASSO)
# Best Lambda Value - Group LASSO
GLASSO.lmbda=log(fit.GLASSO$lambda.min)
# Group LASSO Model Plots
plot(fit.GLASSO$gglasso.fit,
     xlim = c(-6,-3.75),
     ylim = c(-2,2.5))
abline(v = GLASSO.lmbda)
text(-5.8,2.2, bquote(lambda == .(round(GLASSO.lmbda,3))))
plot(coef(fit.GLASSO)[-1],
     ylab = "Coefficient Value",
     xlab = "Variable",
     main = "Coefficient: Group LASSO")
text(44,1.2, "Partial Group 1 Shrinkage", col = "Red")
abline(v=c(51,60), h=0, col = c("Black","Red","Red"))
# Elastic Net Plot
plot(fit.glmN)
# Confusion Matrices
mlp_matrix
LASSO_matrix
glmN_matrix
GLASSO_matrix
RGroup_matrix
```